A method for detecting relevant inputs while satisfying a reachability specification for piecewise affine systems

Riccardo Vignali and Maria Prandini

***Abstract*— We address input design for piecewise affine systems so as to satisfy a reachability specification, and, simultaneously, detect the inputs that are actually relevant (influential inputs). This can be useful for system verification, when one has to check if some undesired/unsafe behavior coded via a reachability specification can actually occur and then possibly take appropriate corrective actions . The proposed method is based on a procedure to assess whether a set of inputs are non-influential for the given specification. This procedure rests on a geometrical set-containment test that involves a projection of the set of states that the system can reach and the set of states that represents the specification. The test can be formulated as a mixed integer feasibility problem and integrated in a depth-first exploration algorithm that returns the maximum number of non-influential inputs.**

I. INTRODUCTION

A reachability problem typically consists in verifying if the state of a system can reach a certain region, starting from some initial condition (*reachability specification*). Reachability analysis can be useful in the context of system verification, when, for example, one has to check if some misbehavior – expressed in terms of a reachability specification – may occur, so as to eventually take an appropriate action and redesign the system. If the system evolution depends on some control inputs, then, reachability analysis becomes an input design problem where one has to verify if the control inputs can be set so as to make the system satisfy the reachability specification. Among the possible solutions to this problem, we look for the one that maximizes the set of inputs that are *non-influential* (i.e., those that can take an arbitrary value in their range without compromising the satisfaction of the specification) so as to detect which are the actually relevant inputs and ease the identification of the cause of the malfunctioning and the task of devising a corrective action.

Reachability analysis has been extensively treated in literature, and effective solutions have been developed for diverse classes of systems. Many of the proposed techniques originate from the *model checking* context, where the aim is typically to check if a given system with no control inputs satisfies a certain specification related to its evolution in time. For discrete systems, it has been shown that checking if a discrete system S satisfies a specification can be done by means of a reachability test on the enlarged system obtained by composing S with the *test automaton* that translates the specification (see e.g. [7]). Model checkers have been developed to solve the resulting reachability test.

The main issue in performing reachability analysis is the ability to "compute" with sets, i.e., to represent sets and propagate them through the system dynamics. In model checkers this process is made fully automatic. In the case of finite discrete systems, sets can be represented by enumeration, and reachable sets can be computed starting from the given initial condition and adding one-step successors till convergence is achieved. Termination of the algorithm is guaranteed since the state space is finite. The technical challenge for the verification of discrete system is to devise algorithms and data structure to handle large state spaces.

For continuous state systems, model checking techniques are trickier to be implemented, since, in principle, they would require to analyze all possible infinite evolutions of the systems. The idea is then to rely on the computation (either exact or approximated) of the reachable set of the system under test, so as to consider in a compact way all its infinite possible evolutions. There is indeed a vast literature on reachability analysis for various classes of systems, mainly deterministic (see [1]–[3], [6], [11], [13]–[17], [19] to name a few), including the study of tools for reach sets computation. Different representations of reach sets (polyhedra, zonotopes, ellipsoids) are adopted. The main idea is to start from sets that are easy to represent in a compact form and approximating the system dynamics so that the sets obtained through the direct evolution of the approximated system admit the same representation of the starting sets, while ensuring over-approximation of the reachable sets of the original system. Other approaches are based on abstracting the continuous system to a discrete one (by means, for example, of a *bisimulation* [5]), and then analyzing the latter via model checking techniques. Many of these methods can be extended to the class of *hybrid* systems, i.e., systems with interacting discrete and continuous dynamics [4], [11], which include the class of PieceWise Affine (PWA) systems.

In this paper we investigate bounded time reachability with relevant input detection for discrete time PWA systems. We addressed the same problem for the class of linear systems in [20], [21] where, inspired by [11], [19], we adopted an optimization-based perspective for maximizing the number of non-influential inputs. Direct extension of the method in [20], [21] to a PWA framework leads to a conservative solution as pointed out in [22]. In this paper, we propose a different approach that overcomes such a conservativeness and rests on a geometrical reinterpretation of the problem.

This work is partially supported by the European Commission under the project UnCoVerCPS with grant number 643921.

R. Vignali and M. Prandini are with Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy {riccardomaria.vignali, maria.prandini}@polimi.it

The core of the approach is the formulation of a suitable test to assess whether or not a set of inputs are non-influential for the given reachability specification. The test reduces to verify a geometrical set-containment condition that involves a projection of the set of states that the system can reach and the set of states that represents the specification. We shall then show how the set-containment condition can be turned into a mixed integer feasibility problem and integrated in a depth-first exploration algorithm that returns the maximum number of non-influential inputs.

The rest of the paper is organized as follows. We describe the class of PWA systems and the proposed geometrical approach in Section II. Section III provides a numerical example and Section IV draws some concluding remarks.

II. Problem formulation and resolution

A PWA system with state $x \in \mathcal{X} \subseteq \mathbb{R}^n$, input $u \in \mathcal{U} \subseteq \mathbb{R}^m$, and output $y \in \mathbb{R}^p$is described by

$$\begin{aligned} x(k+1) &= A_i x(k) + B_i u(k) + f_i \\ y(k) &= C_i x(k) + D_i u(k) + g_i \end{aligned} \quad \text{for} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \mathcal{A}_i, \tag{1}$$

where f_i, g_i are constant vectors and

$$\mathcal{A}_i = \{(x,u) : [L^i_{ax} \; L^i_{au}] \begin{bmatrix} x \\ u \end{bmatrix} \leq L^i_b\}$$

is a polyhedron. System (1) is *well posed* if its evolution is always well-defined. A numerical test for checking if a system is well posed can be found in [10].

Here we assume that the collection $\{\mathcal{A}\}_{i=1}^s$ forms a *polyhedral subdivision* of the space $\mathcal{X} \times \mathcal{U}$.[1] Each polyhedron $\mathcal{A}_i$ represents a *mode* of the PWA system and mode j is *active* at time t if $(x(t), u(t)) \in \mathcal{A}_j$. Note that, since the modes define a partition of the state-input space, there is one and only one active mode at a time. Based the definition of modes, the state evolution of (1) can be compactly written as:

$$x(k) = \prod_{j=0}^{k-1} A_{i(j)} x(0) + \sum_{j=0}^{k-1} \Big(\prod_{h=j+1}^{k-1} A_{i(h)} B_{i(j)} u(j) + f_{i(j)} \Big),$$

where $i(j) \in \{1, \ldots, s\}$ is the index of the mode active at time j. The sequence of modes $I = \{i(0), i(1), \ldots, i(k-1)\} \in \{1, \ldots, s\}^k$ is called *switching sequence*.

We consider the case when the system has multiple scalar inputs ($m > 1$) and each input u_i is bounded in some interval, i.e., $u_i \in [\underline{u}_i, \overline{u}_i]$, $i = 1, \ldots, m$. The input set is then given by $\mathcal{U} = \bigtimes_{i=1}^m [\underline{u}_i, \overline{u}_i]$.

We are now in a position to formally state our input design problem for reachability with relevant input detection.

Let us consider the polytopic set $\mathcal{X}_f = \{x \in \mathcal{X} : H_a x \leq H_b\}$. Our goal is to make the state of the PWA system (1) reach $\mathcal{X}_f$ at some time T while maximizing the number of non-influential inputs.

[1] $\cup_{i=1}^s \mathcal{A}_i = \mathcal{X} \times \mathcal{U}$, each $\mathcal{A}_i$ is of dimension $n+m$, and the intersection $\mathcal{A}_i \cap \mathcal{A}_j$, $i \neq j$, is either empty or a common proper face of both polyhedra.

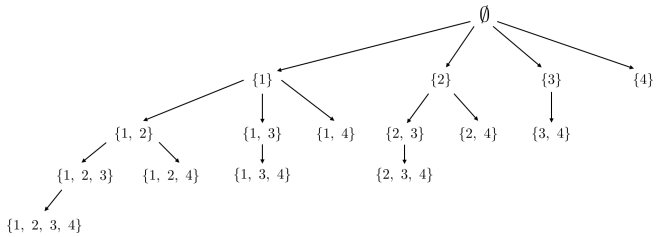

Fig. 1: *Exploration tree of subsets of non-influential inputs for $\mathcal{M} = \{1,\ 2,\ 3,\ 4\}$ by depth-first strategy. If node $\{1,\ 2\}$ is not passing the test, then the subtree originated from node $\{1,\ 2\}$ is removed from the exploration, and so is the branch $\{4\}$ since it does not originate subtrees with cardinality greater than 1.*

The problem can be formulated as follows:

$$\begin{aligned} \max_{\mathcal{N} \in 2^{\mathcal{M}},\ \{u_{j_l}(k)\}_{j_l \in \mathcal{I} = \mathcal{M} \setminus \mathcal{N}}} \quad & |\mathcal{N}| \\ & x(T) \in \mathcal{X}_f, \\ & x(k+1) = A_i x(k) + B_i u(k) + f_i, \quad \text{for} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \mathcal{A}_i \\ & \forall u_{j_h} \in \left[\underline{u}_{j_h},\ \overline{u}_{j_h}\right],\ j_h \in \mathcal{N},\ h = 1, 2, \ldots |\mathcal{N}|, \\ & k = 0, \ldots, T-1 \end{aligned} \tag{2}$$

where $\mathcal{M} = \{1, 2, \ldots, m\}$ represents the set of input indexes, $|\mathcal{C}|$ denotes the cardinality of set $\mathcal{C}$ and $2^{\mathcal{C}}$ is the power set of $\mathcal{C}$.

Problem (2) aims at identifying a sequence for the influential inputs (with indexes in set $\mathcal{I}$) that makes the other inputs (with indexes in set $\mathcal{N}$) non-influential while satisfying the reachability specification. Note that sets $\mathcal{I}$ and $\mathcal{N}$ are not known a-priori, and the aim is to maximize the number of non-influential inputs, i.e., the cardinality of set $\mathcal{N}$.

Problem (2) is in general difficult to solve, since the optimization has to be performed over all subsets $\mathcal{N}$ of $\mathcal{M}$. We then propose to address it via a depth-first exploration strategy that at each step tests whether a subset of inputs are influential or not. Before detailing the approach we introduce the definition of *jointly* influential inputs.

Definition 2.1: Inputs $\{u_{j_h},\ j_h \in \mathcal{N}\}$, $\mathcal{N} \subseteq \mathcal{M}$, are said to be *jointly* non-influential if $\mathcal{N}$ is feasible for problem (2).

Suppose that we have a function $\mathcal{T} : 2^{\mathcal{M}} \to \{0,\ 1\}$, that checks whether the inputs in $\mathcal{N} \in 2^{\mathcal{M}}$ are jointly non-influential or not, and returns 1 or 0 accordingly. In principle, we should run $\mathcal{T}$ for all the subsets $\mathcal{N} \in 2^{\mathcal{M}}$ in order to find the subset with maximal cardinality that passes the test. In practice, we can exploit the following trivial property:

Property 1: Inputs $\{u_j,\ j \in \mathcal{N}\}$, with $\mathcal{N} \subseteq \mathcal{M}$, can be jointly non-influential only if inputs $\{u_j,\ j \in \overline{\mathcal{N}}\}$ are jointly non-influential for any $\overline{\mathcal{N}} \subseteq \mathcal{N}$.

This property suggests to look for the subset of non-influential inputs with maximal cardinality through a depth first exploration strategy. We construct a tree where each node represents a subset of inputs $\mathcal{N}$ that is tested via function $\mathcal{T}$; at each depth d, the nodes correspond to subsets with cardinality d (see Figure 1 with $\mathcal{M} = \{1,\ 2,\ 3,\ 4\}$).

The exploration of the tree starts from the left hand side branch and takes advantage of the property described above, since every time a node fails the test, we can remove the subtree that originates from that node as explained in the caption of Figure 1. Once a node with cardinality $\bar{d}$ passes the test, the subtrees containing modes with cardinality at most $\bar{d}$ are not tested. In the next section we describe how to check whether the inputs in a subset of $\mathcal{M}$ are non-influential.

A. Test function

Consider a switching sequence $I_l = \{i_0,\ i_1, \ldots, i_{T-1}\}$. The evolution of the state of system (1) on the time interval $[0,\ T]$ starting from $x(0) = x_0$ can be written compactly as:

$$X = \mathbf{A}^{(I_l)} x_0 + \mathbf{B}^{(I_l)} U + \mathbf{G}^{(I_l)} F^{(I_l)}, \tag{3}$$

where

$$X = \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(T) \end{bmatrix},\ U = \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ u(T-1) \end{bmatrix},\ F^{(I_l)} = \begin{bmatrix} f_{i_0} \\ f_{i_1} \\ f_{i_2} \\ \vdots \\ f_{i_{T-1}} \end{bmatrix}$$

and $\mathbf{A}^{(I_l)}$, $\mathbf{B}^{(I_l)}$, and $\mathbf{G}^{(I_l)}$ are suitably defined matrices. The constraints on the state-input pairs due to the mode activation in the switching sequence I_l can be written as:

$$\mathbf{L}_{ax}^{(I_l)} X + \mathbf{L}_{au}^{(I_l)} U \leq \mathbf{L}_b^{(I_l)}, \tag{4}$$

where

$$\mathbf{L}_{ax}^{(I_l)} = \begin{bmatrix} L_{ax}^{i_0} & & \\ & \ddots & \\ & & L_{ax}^{i_{T-1}} \end{bmatrix}, \mathbf{L}_{au}^{(I_l)} = \begin{bmatrix} L_{au}^{i_0} & & \\ & \ddots & \\ & & L_{au}^{i_{T-1}} \end{bmatrix},$$

$$\mathbf{L}_b^{(I_l)} = \left[L_b^{i_0\prime},\ L_b^{i_1\prime},\ \ldots L_b^{i_{T-1}\prime} \right]'.$$

By plugging (3) in (4) we get:

$$\mathbf{L}_{ax}^{(I_l)} (\mathbf{A}^{(I_l)} x_0 + \mathbf{B}^{(I_l)} U + \mathbf{G}^{(I_l)} F^{(I_l)}) + \mathbf{L}_{au}^{(I_l)} U \leq \mathbf{L}_b^{(I_l)},$$

which becomes $\mathbf{M}_a^{(I_l)} U \leq \mathbf{M}_b^{(I_l)}$, where we set $\mathbf{M}_a^{(I_l)} = \mathbf{L}_{ax}^{(I_l)} \mathbf{B}^{(I_l)} + \mathbf{L}_{au}^{(I_l)}$ and $\mathbf{M}_b^{(I_l)} = \mathbf{L}_b^{(I_l)} - \mathbf{L}_{ax}^{(I_l)} \mathbf{A}^{(I_l)} x_0 - \mathbf{L}_{ax}^{(I_l)} \mathbf{G}^{(I_l)} F^{(I_l)}$. Note that it holds that $\bigcup_{I_l \in \mathbf{SS}} \boldsymbol{\mathcal{A}}_l = \mathcal{U}^T$, where

$$\boldsymbol{\mathcal{A}}_l = \left\{ U \in \mathcal{U}^T : \mathbf{M}_a^{(I_l)} U \leq \mathbf{M}_b^{(I_l)} \right\}$$

and $\mathbf{SS}$ denotes the set of all possible switching sequences. This is because at each time step k it is possible to inject in the system any input vector $u(k) \in \mathcal{U}$, since there always exists an active mode corresponding to that input due to the polyhedral subdivision of $\mathcal{X} \times \mathcal{U}$, and the system evolution is in fact well defined. The reachability constraint can be rewritten as:

$$H_a (\mathbf{A}_T^{(I_l)} x_0 + \mathbf{B}_T^{(I_l)} U + \mathbf{G}_T^{(I_l)} F^{(I_l)}) \leq H_b, \tag{5}$$

where $\mathbf{A}_T^{(I_l)}$, $\mathbf{B}_T^{(I_l)}$ and $\mathbf{G}_T^{(I_l)}$ are obtained by extracting the last n rows of matrices $\mathbf{A}^{(I_l)}$,$\mathbf{B}^{(I_l)}$ and $\mathbf{G}^{(I_l)}$, respectively. We can rewrite equation (5) in the compact form:

$$\mathbf{S}_a^{(I_l)} U \leq \mathbf{S}_b^{(I_l)},$$

where $\mathbf{S}_a^{(I_l)} = H_a \mathbf{B}_T^{(I_l)}$ and $\mathbf{S}_b^{(I_l)} = H_b - H_a \mathbf{A}_T^{(I_l)} x_0 - H_a \mathbf{G}_T^{(I_l)} F^{(I_l)}$, and define the set

$$\boldsymbol{\mathcal{S}p}_l = \{ U \in \mathcal{U}^T : \mathbf{S}_a^{(I_l)}\ U \leq \mathbf{S}_b^{(I_l)} \}.$$

If the set $\boldsymbol{\mathcal{A}}_l \cap \boldsymbol{\mathcal{S}p}_l$ is not empty, its elements are input sequences $\{u(0),\ u(1), \ldots u(T-1)\}$ that drive the state of the system into the target set $\mathcal{X}_f$ by keeping its evolution in the switching sequence I_l.

We now discuss the formulation of function $\mathcal{T}$ by first considering the case $\mathcal{N} = \mathcal{M}$ and then the case $\mathcal{N} \subset \mathcal{M}$.

Case $\mathcal{N} = \mathcal{M}$: In this case we want to test if all the inputs of system (1) are jointly non-influential. Clearly, this is true if and only if all the possible evolutions of the system starting from the given initial condition satisfy the specification. This means that for any possible choice of the inputs in the enlarged space $\mathcal{U}^T$, the specification have to be satisfied, which, thanks to the property of the sets $\boldsymbol{\mathcal{A}}_l$ outlined above, can be equivalently stated as:

$$\boldsymbol{\mathcal{A}}_l \subseteq \boldsymbol{\mathcal{S}p}_l \quad \forall I_l \in \mathbf{SS}. \tag{6}$$

Provided that both the sets $\boldsymbol{\mathcal{A}}_l$ and $\boldsymbol{\mathcal{S}p}_l$ are polyhedra, condition $\boldsymbol{\mathcal{A}}_l \subseteq \boldsymbol{\mathcal{S}p}_l$ can be checked via the following linear feasibility test, derived from standard dual optimization arguments:

$$\begin{aligned} \min_{\mathcal{H}^{(I_l)}}\ & 0 \\ & \mathcal{H}^{(I_l)} \mathbf{M}_a^{(I_l)} = \mathbf{S}_a^{(I_l)} \\ & \mathcal{H}^{(I_l)} \mathbf{M}_b^{(I_l)} \leq \mathbf{S}_b^{(I_l)} \\ & \mathcal{H}^{(I_l)} \geq 0 \end{aligned} \tag{7}$$

where $\mathcal{H}^{(I_l)}$ is a matrix of Lagrangian multipliers. Proposition 1 then follows immediately.

Proposition 1: All the inputs of system (1) are jointly non-influential if and only if problem (7) is feasible for all switching sequences.

Note that, even if the single feasibility test (7) is in general not hard to solve, we have to perform it for all the possible switching sequences in $\mathbf{SS}$, which can be computationally demanding. Nonetheless, removing the need of analyzing all the switching sequences seems to be very hard, if not impossible, in the general case: indeed, by detecting the non-influential inputs, we are implicitly assessing a property of the reach set of the PWA system, and thus the analysis of all its possible different behaviors seems unavoidable.

Case $\mathcal{N} \subset \mathcal{M}$: A condition similar to (6) can be also derived for the case of $\mathcal{N} \subset \mathcal{M}$. To this end we denote with $\Pi_{\tilde{u}}^{\mathcal{I}}(\boldsymbol{\mathcal{A}}_l)$ the slice of set $\boldsymbol{\mathcal{A}}_l$ obtained by fixing in U the sequence $\{u_i(k)\}_{k=0}^{T-1}$, $i \in \mathcal{I} = \mathcal{M} \setminus \mathcal{N}$ to the value $\tilde{u} \in \bigtimes_{i \in \mathcal{I}} [\underline{u}_i,\ \overline{u}_i]$. Note that it may be the case that the slice is an empty set (see Figure 2). The inputs in the set $\mathcal{N}$ are jointly non-influential if there exists some value for the influential inputs $\tilde{u} = \{u_i(k)\}_{k=0}^{T-1}$, $i \in \mathcal{I}$, such that the

following condition holds:

$$\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{A}}_l\right) \subseteq \Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{S}}\boldsymbol{p}_l\right) \quad \forall I_l \in \mathbf{SS}. \tag{8}$$

Note that (8) is trivially satisfied when $\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{A}}_l\right) = \emptyset$. By the definitions of $\boldsymbol{\mathcal{A}}_l$ and $\boldsymbol{\mathcal{S}}\boldsymbol{p}_l$ we have that:

$$\begin{aligned}\Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{A}}_l) &= \left\{U_j \in \bigtimes_{j\in\mathcal{N}} [\underline{u}_j,\ \overline{u}_j]^T : \left[\mathbf{M}_a^{(I_l)}\right]^{\mathcal{N}} U_j \leq \mathbf{M}_b^{(I_l)} - \left[\mathbf{M}_a^{(I_l)}\right]^{\mathcal{I}} \tilde{u}\right\} \\ \Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{S}}\boldsymbol{p}_l) &= \left\{U_j \in \bigtimes_{j\in\mathcal{N}} [\underline{u}_j,\ \overline{u}_j]^T : \left[\mathbf{S}_a^{(I_l)}\right]^{\mathcal{N}} U_j \leq \mathbf{S}_b^{(I_l)} - \left[\mathbf{S}_a^{(I_l)}\right]^{\mathcal{I}} \tilde{u}\right\}\end{aligned} \tag{9}$$

where we denote with $[Z]^{\mathcal{N}}$ the matrix obtained by extracting from Z the columns that correspond to the inputs in $\mathcal{N}$ (i.e., columns $\{j, j+m, j+2m, \ldots, j+(T-1)m\}, \forall j \in \mathcal{N}$). Clearly, the sets $\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{A}}_l\right)$ and $\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{S}}\boldsymbol{p}_l\right)$ are polyhedra as well. By following the same reasoning of the case $\mathcal{N} = \mathcal{M}$, it can be shown that the set-containment condition (8) can be checked via the following feasibility test:

$$\begin{aligned}\min_{\tilde{u} \in \underset{i\in\mathcal{I}}{\times}[\underline{u}_i,\ \overline{u}_i]^T,\ \{\mathcal{H}^{(I_l)}\}_{I_l\in\mathbf{SS}}} \quad & 0 \\ \mathcal{H}^{(I_l)} \left[\mathbf{M}_a^{(I_l)}\right]^{\mathcal{N}} &= \left[\mathbf{S}_a^{(I_l)}\right]^{\mathcal{N}} \\ \mathcal{H}^{(I_l)} \left(\mathbf{M}_b^{(I_l)} - \left[\mathbf{M}_a^{(I_l)}\right]^{\mathcal{I}} \tilde{u}\right) &\leq \mathbf{S}_b^{(I_l)} - \left[\mathbf{S}_a^{(I_l)}\right]^{\mathcal{I}} \tilde{u} \\ \mathcal{H}^{(I_l)} &\geq 0 \\ \forall I_l &\in \mathbf{SS}\end{aligned} \tag{10}$$

which is bilinear because of the presence of products between optimization variables. A feasible solution to problem (10) is an instance of the influential inputs i, with $i \in \mathcal{I}$, that slices each set $\boldsymbol{\mathcal{A}}_l$ and $\boldsymbol{\mathcal{S}}\boldsymbol{p}_l$ in the subsets $\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{A}}_l\right)$ and $\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{S}}\boldsymbol{p}_l\right)$, such that $\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{A}}_l\right) \subseteq \Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{S}}\boldsymbol{p}_l\right)$, $\forall I_l \in \mathbf{SS}$.

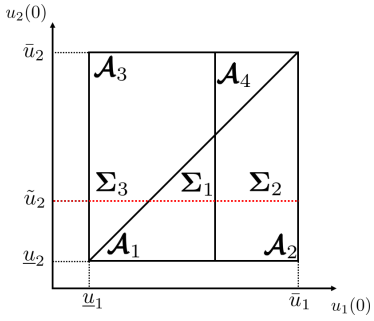

Fig. 2: *Graphical representation of the enlarged input space $\mathcal{U}^T$ in the case of $m = 2$ and $T = 1$. Fixing the input u_2 to the value $\tilde{u}_2$ slices the sets $\boldsymbol{\mathcal{A}}_1$, $\boldsymbol{\mathcal{A}}_2$, $\boldsymbol{\mathcal{A}}_3$ and determines the sets $\boldsymbol{\Sigma}_1$, $\boldsymbol{\Sigma}_2$, $\boldsymbol{\Sigma}_3$. Set $\boldsymbol{\mathcal{A}}_4$ has no intersection with $u_2(0) = \tilde{u}_2$ so that $\boldsymbol{\Sigma}_4 = \emptyset$. If it is the case that $\Sigma_i \subseteq \boldsymbol{\mathcal{S}}\boldsymbol{p}_l,\ i = 1, 2, 3, 4$, then the value $\tilde{u}_2$ is an appropriate choice for u_2 to make input u_1 non-influential. Given that $\bigcup_{I_l\in\mathbf{SS}} \boldsymbol{\mathcal{A}}_l = \mathcal{U}^T$, we are guaranteed that $\cup_{i=1}^4 \boldsymbol{\Sigma}_i = [\underline{u}_1, \overline{u}_1]$*

For a high dimensional problems, existing methods like those in [18], [12] for solving bilinear feasibility tests are not applicable in practice and (10) turns out to be untractable. Therefore, in the following, we show how to formulate a tractable approximation of problem (10) via Mixed Integer Linear Programming.

The bilinearity of problem (10) is caused by the fact that the right-hand-side of the inequality defining polyhedron $\Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{A}}_l)$ is not known a priori, but have to be properly set (see (9)). If set $\boldsymbol{\mathcal{A}}_l$, once sliced, generates a set whose "shape" is independent of where it is sliced, then the right-hand-side would not depend on $\tilde{u}$. This can be actually attained when the set $\boldsymbol{\mathcal{A}}_l$ is a box. In this case, the set obtained by slicing $\boldsymbol{\mathcal{A}}_l$ would be either a box (if $\tilde{u}$ is chosen inside $\boldsymbol{\mathcal{A}}_l$) or the empty set (if $\tilde{u}$ is chosen outside $\boldsymbol{\mathcal{A}}_l$). In general, when $\boldsymbol{\mathcal{A}}_l$ is not a box, it can be overapproximated by an outer box $\boldsymbol{\mathcal{B}}_l$ so as to provide a sufficient condition for (8) to hold, i.e.,

$$\Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{B}}_l) \subseteq \Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{S}}\boldsymbol{p}_l) \Rightarrow \Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{A}}_l) \subseteq \Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{S}}\boldsymbol{p}_l). \tag{11}$$

Naturally, we choose $\boldsymbol{\mathcal{B}}_l$ to be the *minimum volume* outer box approximation of $\boldsymbol{\mathcal{A}}_l$. If we denote with $\bar{b}^{(I_l)}$ and $\underline{b}^{(I_l)}$ the upper and lower limits of $\boldsymbol{\mathcal{B}}_l$ we have that:

$$\boldsymbol{\mathcal{B}}_l = \{U \in \mathcal{U}^T : \mathbf{B}_a^{(I_l)} U \leq \mathbf{B}_b^{(I_l)}\} \tag{12}$$

where $\mathbf{B}_a^{(I_l)} = [\mathbf{I}_{mT},\ \ -\mathbf{I}_{mT}]'$, $\mathbf{B}_b^{(I_l)} = [\bar{b}^{(I_l)'},\ \ -\underline{b}^{(I_l)'}]'$, $\mathbf{I}_{mT}$ is the identity matrix with mT dimensions and the mT elements of the vectors $\bar{b}^{(I_l)}$ and $\underline{b}^{(I_l)}$ are obtained as in [19]. The set $\Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{B}}_l)$ obtained by slicing $\boldsymbol{\mathcal{B}}_l$ in correspondence of $\tilde{u}$ can be written as:

$$\Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{B}}_l) = \left\{U \in \bigtimes_{j\in\mathcal{N}} [\underline{u}_j,\ \overline{u}_j]^T : \left[\mathbf{B}_a^{(I_l)}\right]^{\mathcal{N}} U \leq \mathbf{B}_b^{(I_l)} - \left[\mathbf{B}_a^{(I_l)}\right]^{\mathcal{I}} \tilde{u}\right\}$$

or, alternatively,

$$\Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{B}}_l) = \begin{cases} \bigtimes_{k=0}^{T-1} \bigtimes_{j\in\mathcal{N}} [\underline{b}_j^{(I_l)}(k), \bar{b}_j^{(I_l)}(k)], \tilde{u} \in \bigtimes_{k=0}^{T-1} \bigtimes_{i\in\mathcal{I}} [\underline{b}_i^{(I_l)}(k), \bar{b}_i^{(I_l)}(k)] \\ \emptyset, \qquad \text{otherwise} \end{cases}.$$

For ease of notation, we denote with $\underline{b}_{\mathcal{N}}^{(I_l)}$ and $\bar{b}_{\mathcal{N}}^{(I_l)}$ the lower and upper limits of the sliced box, obtained by stacking in column the scalar elements of $\underline{b}^{(I_l)}$ and $\bar{b}^{(I_l)}$ that correspond to the inputs in $\mathcal{N}$. The case of the sliced box $\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{B}}_l\right)$ being non-empty is addressed in the following proposition whose proof is omitted due to space limitations (see [22]).

Proposition 2: Let $\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{B}}_l\right) \neq \emptyset$. Then, the condition $\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{B}}_l\right) \subseteq \Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{S}}\boldsymbol{p}_l\right)$ is satisfied if and only if:

$$\exists \tilde{u} \in [\underline{b}_{\mathcal{N}}^{(I_l)} \quad \bar{b}_{\mathcal{N}}^{(I_l)}] : \quad \left[\mathbf{S}_a^{(I_l)}\right]^{\mathcal{I}} \tilde{u} \leq \mathbf{S}_b^{(I_l)} - \mathcal{L}^{(I_l)}, \tag{13}$$

with $\mathcal{L}^{(I_l)} = \frac{1}{2}\left[\mathbf{S}_a^{(I_l)}\right]^{\mathcal{N}} (\bar{b}_{\mathcal{N}}^{(I_l)} + \underline{b}_{\mathcal{N}}^{(I_l)}) + \frac{1}{2}\left|\left[\mathbf{S}_a^{(I_l)}\right]^{\mathcal{N}}\right| (\bar{b}_{\mathcal{N}}^{(I_l)} - \underline{b}_{\mathcal{N}}^{(I_l)})$.

Proposition 2 states that a value for $\tilde{u}$ that makes the set containment condition (8) hold can be found via a linear feasibility test. In general, we would like to solve the feasibility test (13) for all the possible switching sequences, so that the set containment condition is satisfied for all of them. Nonetheless, we can not simply stack all the constraints of the form (13) for every possible switching sequence, since this would lead to an unfeasible problem. In fact, in general, there is no common value for $\tilde{u}$ shared by all the switching sequences, so that some of the sets $\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{B}}_l\right)$ may result to be empty. For these sets we would like to make the condition $\Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{B}}_l\right) \subseteq \Pi^{\mathcal{I}}_{\tilde{u}}\left(\boldsymbol{\mathcal{S}}\boldsymbol{p}_l\right)$ trivial, so as to guarantee the feasibility of the whole problem. For this reason, we split the condition

$\Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{B}}_l) \subseteq \Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{S}p}_l)$ in the following two parts:

- if $\tilde{u} \in [\underline{b}^{(I_l)}_{\mathcal{I}},\ \bar{b}^{(I_l)}_{\mathcal{I}}]$, $\Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{B}}_l) \subseteq \Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{S}p}_l) \Leftrightarrow \left[\mathbf{S}^{(I_l)}_a\right]^{\mathcal{I}} \tilde{u} \leq \mathbf{S}^{(I_l)}_b - \mathcal{L}^{(I_l)}$,
- if $\tilde{u} \notin [\underline{b}^{(I_l)}_{\mathcal{I}},\ \bar{b}^{(I_l)}_{\mathcal{I}}]$, $\Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{B}}_l) \subseteq \Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{S}p}_l)$ is always satisfied

We now show how to encode these conditions in a Mixed Integer Feasibility Test. Consider a single switching sequence: the constraint expressing the condition that $\tilde{u}$ belongs to the box $\boldsymbol{\mathcal{B}}_l$ can be expressed as:

$$\mathbf{T}_a \tilde{u} \leq \mathbf{T}^{(I_l)}_b, \tag{14}$$

where $\mathbf{T}_a = [\mathbf{I}_{|\mathcal{I}|T},\ -\mathbf{I}_{|\mathcal{I}|T}]'$ and $\mathbf{T}^{(I_l)}_b = [\bar{b}^{(I_l)\prime}_{\mathcal{I}} - \underline{b}^{(I_l)\prime}_{\mathcal{I}}]'$. We now introduce $2|\mathcal{I}|T$ binary variables $\sigma^{(I_l)}_t$, $t = \{1, \ldots, 2|\mathcal{I}|T\}$, one per each row of (14), defined by $\sigma^{(I_l)}_t = 1 \Leftrightarrow \mathbf{T}_{a,t}\tilde{u} \leq \mathbf{T}^{(I_l)}_{b,t}$. This definition can be translated into the following linear inequalities via the *big-M* technique, [8]:

$$\begin{aligned} \mathbf{T}_{a,t}\tilde{u} - \mathbf{T}^{(I_l)}_{b,t} + \bar{V}^{(I_l)}_t \sigma^{(I_l)}_t &\leq \bar{V}^{(I_l)}_t \\ -\mathbf{T}_{a,t}\tilde{u} + \mathbf{T}^{(I_l)}_{b,t} + \underline{V}^{(I_l)}_t \sigma^{(I_l)}_t &\leq 0, \end{aligned}$$

where $\bar{V}^{(I_l)}_t = \max_{\tilde{u}} \mathbf{T}_{a,t}\tilde{u} - \mathbf{T}^{(I_l)}_{b,t}$ and $\underline{V}^{(I_l)}_t = \min_{\tilde{u}} \mathbf{T}_{a,t}\tilde{u} - \mathbf{T}^{(I_l)}_{b,t}$ can be computed by using the same technique as in Proposition 2.

In order to make the constraint (13) trivially satisfied when $\tilde{u}$ is not inside the box $\boldsymbol{\mathcal{B}}_l$ we can now exploit the binary variables defined above and write the following constraint:

$$\left[\mathbf{S}^{(I_l)}_a\right]^{\mathcal{I}} \tilde{u} + \mathbf{Z}^{(I_l)}\sigma^{(I_l)} \leq \mathbf{S}^{(I_l)}_b - \mathcal{L}^{(I_l)} + \mathbf{Z}^{(I_l)}\mathbf{1}_{2|\mathcal{I}|T}, \tag{15}$$

where $\mathbf{1}_{2|\mathcal{I}|T}$ is a column vector of $2|\mathcal{I}|T$ ones, and each column $[\mathbf{Z}]^t$ of $\mathbf{Z}$ is defined by:

$$[\mathbf{Z}]^t = \max_{\tilde{u}} [\mathbf{S}^{(I_l)}_a]^{\mathcal{I}} \tilde{u} - \mathbf{S}^{(I_l)}_b + \mathcal{L}^{(I_l)}, \tag{16}$$

where the max operator is to be interpreted row-wise. Again, (16) can be tackled by means of the same technique used in Proposition 2. Equation (15) becomes tight only if *all* the $\sigma^{(I_l)}_t$, $t = 1, \ldots, 2|\mathcal{I}|T$, are equal to 1, which is equivalent to say that $\tilde{u}$ belongs to the box $\boldsymbol{\mathcal{B}}_l$.

Finally, we can setup the following Mixed Integer Feasibility Test:

$$\begin{aligned} \min_{\tilde{u},\ \{\sigma^{I_l}\}_{I_l \in \mathbf{SS}}} \quad & 0 \\ & \mathbf{T}_{a,t}\tilde{u} - \mathbf{T}^{(I_l)}_{b,t} + \bar{V}^{(I_l)}_t \sigma^{(I_l)}_t \leq \bar{V}^{(I_l)}_t \\ & -\mathbf{T}_{a,t}\tilde{u} + \mathbf{T}^{(I_l)}_{b,t} + \underline{V}^{(I_l)}_t \sigma^{(I_l)}_t \leq 0 \\ & \left[\mathbf{S}^{(I_l)}_a\right]^{\mathcal{I}} \tilde{u} + \mathbf{Z}^{(I_l)}\sigma^{(I_l)} \leq \mathbf{S}^{(I_l)}_b - \mathcal{L}^{(I_l)} + \mathbf{Z}^{(I_l)}\mathbf{1}_{2|\mathcal{I}|T} \\ & \forall I_l \in \mathbf{SS}. \end{aligned} \tag{17}$$

If feasible, problem (17) returns a sequence $\tilde{u} = \{u_i(0), \ldots, u_i(T-1)\}$, $i \in \mathcal{I}$ such that all the non-empty slices $\Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{B}}_l)$ of the sets $\boldsymbol{\mathcal{B}}_l$ are entirely contained in the corresponding slices $\Pi^{\mathcal{I}}_{\tilde{u}}(\boldsymbol{\mathcal{S}p}_l)$ of the sets $\boldsymbol{\mathcal{S}p}_l$, which implies that the inputs in $\mathcal{N} = \mathcal{M} \setminus \mathcal{I}$ are jointly non-influential.

III. Numerical example

We consider the system in Figure 3 that represents a simplified version of the *input consolidation* scheme of the autopilot of an helicopter. The system has the role of validating the data received by the sensors and guarantees that the measures used in the control loops are reliable. In particular, the validation is performed by introducing two sensors per quantity to be measured (that in this case are the airspeed and the height of the helicopter), and comparing the absolute value of the difference of the data received from the sensors with a threshold. If a pair of sensors passes the test, the corresponding counter is incremented by 1, otherwise it is reset to 0. If the counter exceeds a predefined threshold, here set to 2.5, (i.e., the corresponding sensors measures are close for 3 consecutive time steps), then the counter keeps that value even if the data keep passing the test in the next time steps. Note that the threshold on the values measured by the sensors is a user-modifiable variable in the test phase, and is therefore considered as an additional input. The resulting system has 2 state variables – `AIRSP_COUNTER_PREVIOUS` and `RHT_COUNTER_PREVIOUS` – representing the value stored in the counters, and 6 inputs – `AIRSPEED_1`, `AIRSPEED_2`, `RADAR_HEIGHT_1`, `RADAR_HEIGHT_2`, `A_TRESHOLD`, `R_TRESHOLD` – representing the values of two pairs of sensors and the two corresponding thresholds. For ease of notation we will refer to the state variables as x_1 and x_2 and the input signals as u_i, $i = 1, \ldots, 6$.

The range for the inputs are set as follows $u_1 \in [0,\ 160]$, $u_2 \in [0,\ 160]$, $u_3 \in [\epsilon,\ 200]$, $u_4 \in [5,\ 100]$, $u_5 \in [5,\ 100]$, $u_6 \in [\epsilon,\ 200]$, where $\epsilon = 10^{-5}$ is the tolerance above which a number is considered positive in our implementation. We consider the problem of detecting non-influential inputs for the specification: $x_1(T) > 2.5 \ \wedge \ x_2(T) > 2.5$, which is satisfied if for 3 consecutive time instants both pairs of inputs pass the threshold test. Running our algorithm returns the following output:

```
  Computing non-influential inputs for
system:
INP_CONSOL.pwa
Specification: x_1(T)>2.5 ∧ x_2(T)>2.5
Time T -> 3.
... Tree exploration ...
 1) I = 1 2 3 4 5 6   N = * * * * * *
    Time elapsed: 0 h 0 m 17 s -> OK
 2) I = * 2 3 4 5 6   N = 1 * * * * *
    Time elapsed: 0 h 0 m 16 s -> OK
 3) I = * * 3 4 5 6   N = 1 2 * * * *
    Time elapsed: 0 h 0 m 2 s -> OK
 4) I = * * * 4 5 6   N = 1 2 3 * * *
    Time elapsed: 0 h 0 m 12 s -> NO
 5) I = * * 3 * 5 6   N = 1 2 * 4 * *
    Time elapsed: 0 h 0 m 2 s -> OK
 6) I = * * 3 * * 6   N = 1 2 * 4 5 *
    Time elapsed: 0 h 0 m 2 s -> OK
 7) I = * * 3 * * *   N = 1 2 * 4 5 6
    Time elapsed: 0 h 0 m 3 s -> NO
 8) I = * 2 * 4 5 6   N = 1 * 3 * * *
    Time elapsed: 0 h 1 m 4 s -> NO
 9) I = 1 * 3 4 5 6   N = * 2 * * * *
    Time elapsed: 0 h 0 m 16 s -> OK
```

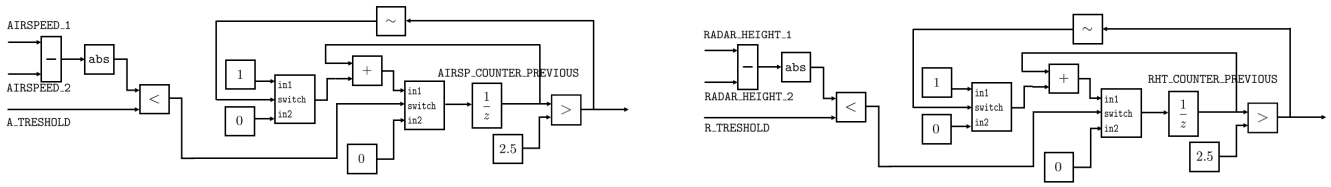

Fig. 3: Input consolidation scheme

```
10) I = * * * 4 5 6   N = * 2 3 * * *
    Time elapsed: 0 h 1 m 16 s -> NO
...done.
Non-influential inputs indexes: 1 2 4 5.
Influential inputs sequence:
u_3 = 160.0001 160.0001 160.0001 160.0001
u_6 = 95.0001  95.0001  95.0001  95.0001
```

where `I` and `N` respectively denote the subsets of influential and non-influential inputs that are checked at each step of the tree exploration. Note that at iteration 7 the maximum number of non-influential inputs found is 4, but there are still two branches (those starting from nodes `{1 * 3 * * *}` and `{* 2 * * * *}`) that can lead to solutions with 5 non-influential inputs; for this reason, the exploration continues and stops only when the possibility of detecting 5 non-influential inputs has been ruled out. We obtain that the only two influential inputs are the thresholds on the sensors measures, u_3, u_6. This is due to the fact that both u_3 and u_6 can be set to a value that is greater than the maximum absolute difference of the corresponding other inputs, i.e., $\overline{u}_3 = 200 > \max_{u_1,\ u_2 \in [0,\ 160]^2} |u_1 - u_2|$ and $\overline{u}_6 = 200 > \max_{u_4,\ u_5 \in [0,\ 100]^2} |u_4 - u_5|$, so that the condition of input validity is always satisfied, for any choice of u_1, u_2, u_4, u_5. Changing the ranges for the threshold inputs can modify this outcome. If we set $(u_3,\ u_4) \in [\epsilon,\ 50]^2$, then the inputs u_3 and u_6 are non-influential. This result is achieved by setting the corresponding pair of inputs u_1, u_2 and u_4, u_5 to the same value at each time instant, so that $|u_1(k) - u_2(k)| = |u_4(k) - u_5(k)| = 0$, $\forall k = 0, \ldots, 3$, and u_3 and u_6 can be therefore set to any positive value.

IV. Conclusions

We proposed a method for detecting relevant inputs in PWA system verification. The resulting MILP problem can be computationally demanding and suitable methods are needed to cope with large scale systems. The joint use of MILP classical solvers and techniques borrowed from theoretical computer science, like boolean satisfiability problem solvers (SAT), can be explored to this end (see [9]).

References

[1] M. Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Hybrid Systems: Computation and Control*, pages 173–182, 2013.

[2] M. Althoff and B. H. Krogh. Reachability analysis of nonlinear differential-algebraic systems. *IEEE Transactions on Automatic Control*, 59(2):371–383, 2014.

[3] M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233–249, 2010.

[4] R. Alur. Formal verification of hybrid systems. In *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*, pages 273–278. IEEE, 2011.

[5] R. Alur, T. Henzinger, G. Lafferriere, G.J. Pappas, et al. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

[6] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *Hybrid Systems: Computation and Control*, pages 20–31. Springer, 2000.

[7] C. Baier and J-P Katoen. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

[8] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and Controllability of Piecewise Affine and Hybrid Systems. *IEEE Transactions on Automatic Control*, 45(10):1864–1876, 2000.

[9] A. Bemporad and N. Giorgetti. Logic-based solution methods for optimal control of hybrid systems. *Automatic Control, IEEE Transactions on*, 51(6):963–976, 2006.

[10] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[11] A. Bemporad and M. Morari. Verification of hybrid systems via mathematical programming. In *Hybrid Systems: Computation and Control*, pages 31–45. Springer, 1999.

[12] A. Costa and L. Liberti. Relaxations of multilinear convex envelopes: dual is better than primal. In *Experimental Algorithms*, pages 87–98. Springer, 2012.

[13] G. Frehse, R. Kateja, and C. Le Guernic. Flowpipe approximation and clustering in space-time. In *Hybrid Systems: Computation and Control*, pages 203–212, 2013.

[14] A. Girard, C. Le Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control*, pages 257–271. Springer, 2006.

[15] C. Le Guernic and A. Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010.

[16] A.A. Kurzhanskiy and P. Varaiya. Ellipsoidal techniques for reachability analysis of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 52(1):26–38, Jan 2007.

[17] I. Mitchell and C.J. Tomlin. Level set methods for computation in hybrid systems. In *Hybrid Systems: Computation and Control*, pages 310–323. Springer, 2000.

[18] H.D. Sherali and A. Alameddine. A new reformulation-linearization technique for bilinear programming problems. *Journal of Global optimization*, 2(4):379–410, 1992.

[19] F.D. Torrisi. *Modeling and reach-set computation for analysis and optimal control of discrete hybrid automata*. PhD thesis, Diss., Technische Wissenschaften ETH Zürich, Nr. 15064, 2003, 2003.

[20] R. Vignali, L. Deori, and M. Prandini. Control input design: detecting non influential inputs while satisfying a reachability specification. In *19th World Congress of the International Federation of Automatic Control*, Cape Town, South Africa, August 2014.

[21] R. Vignali and M. Prandini. Input design for a cascading system: An approach based on system decomposition and non-influential input detection. In *2014 IEEE Multi-conference on Systems and Control*, Antibes, France, October 2014.

[22] R.M. Vignali. *Automatic verification and input design for dynamical systems: an optimization-based approach to the detection of non-influential inputs*. PhD thesis, Politecnico di Milano, Milano, Italy, 2015.